Bayesian learning Methods

Bayesian learning Methods are relevant to our study of machine learning for two different reasons:

① Bayesian learning algorithms that calculate explicit probabilities for hypotheses, such as Naive Bayes Classifier, are among the most practical approaches to certain types of learning problems

② The second reason is that Bayesian methods provide a useful perspective for understanding many learning algorithms that do not explicitly manipulate probabilities.

Two roles for Bayesian methods:

① Provide practical learning algorithms
- Naive Bayes learning ⟶ each observed training example can incrementally decrease or increase the estimated probability that a hypothesis is correct. This provides a more flexible approach to learning than algorithms that completely eliminating a hypothesis if it is found to be inconsistent with any single example.
- Combine prior knowledge (prior probabilities) with observed data;
  ⟶ prior knowledge can be combined with observed data to determine the final probability of a hypothesis;
- Make probabilistic predictions
  ⟶ Bayesian methods can accomodate hypotheses that make probabilistic predictions. New instances can be classified by combining the predictions of multiple hypotheses, weighted by their probabilities;

② Provide useful conceptual framework
- Provide a standard for evaluating other learning algorithms
  ⟶ even in cases where Bayesian methods prove computationally intractable, they can provide a standard of optimal decision making against which other practical methods can be measured

One practical difficulty in applying Bayesian methods is that they typically require initial knowledge of many probabilities. When these probabilities are not known in advance they are often estimated based on background knowledge, previously available data and assumptions about the form of the underlying distributions. A second, practical difficulty is the significant computational cost required to determine the Bayes optimal hypothesis in the general case.

Basic formulas for probabilities

Product rule: probability of conjunction of A and B

$$P(A \wedge B) = P(A|B)P(B) = P(B|A)P(A)$$

Sum rule: probability of disjunction of A and B

$$P(A \vee B) = P(A) + P(B) - P(A \wedge B)$$

Theorem of total probability: if events $A_1, \dots, A_n$ are mutually exclusive with $\sum_{i=1}^{n} P(A_i) = 1$, then

$$P(B) = \sum_{i=1}^{n} P(B|A_i)P(A_i)$$

Bayes theorem: provides a way to calculate posterior probability $P(h|D)$ from the prior probability $P(h)$ together with $P(D)$ and $P(D|h)$

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

In machine learning we are often interested in determining the best hypothesis from some space H, given the observed training data D. One way to specify what we mean by the best hypothesis is to say that we demand the most probable one, given the data D plus any initial knowledge about the prior probabilities of the various hypotheses in H. Bayes theorem provides a direct method for computing such probabilities.

We want to learn a function with a dataset with the goal of classifying a new instance
Given a target function $f: X \to V$, a dataset D and a new instance $x' \notin D$, the best prediction $\hat{f}(x') = v^*$ is:

$$v^* = \underset{v \in V}{\operatorname{argmax}} \; P(v \mid x', D)$$

In other words, given the dataset D and a new sample $x' \notin D$, we want to compute the probability distribution over V: for each possible outcome of $f: X \to V$, we compute the likelihood that it is the right one and take the sample with the higher probability.

$P(V \mid x', D)$ ← probability distribution on the image of f (V)

Given a dataset D and hypothesis space H, compute a probability distribution over H given D.

$$P(H \mid D)$$

We could use the Bayes rule if H is finite and small

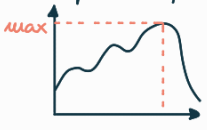
$$P(h \mid D) = \frac{P(D \mid h)\, P(h)}{P(D)}$$

↑ for each h

$P(h)$ = prior probability of hypothesis h
$P(D)$ = prior probability of training data D
$P(h \mid D)$ = probability of h given D – unknown
$P(D \mid h)$ = probability of D given h

Maximum A Posteriori Hypothesis h_{MAP}

In many learning scenarios, the learner consider some set of candidate hypotheses H and is interested in finding the most probable hypothesis $h \in H$ given the observed data D. Any such maximally probable hypothesis is called maximum a posteriori (MAP) hypothesis. We can determine the MAP hypotheses by using Bayes theorem to compute the posterior probability for each candidate.



$$h_{MAP} \triangleq \underset{h \in H}{\operatorname{argmax}} \; P(h \mid D) = \underset{h \in H}{\operatorname{argmax}} \; \frac{P(D \mid h)\, P(h)}{P(D)} = \underset{h \in H}{\operatorname{argmax}} \; P(D \mid h)\, P(h)$$

↳ argmax is invariant with respect to a scaling factor ($1/P(D)$) which is constant with respect to h

Maximum Likelihood Hypotheses

$P(D \mid h)$ is often called the likelihood of the data D given h and any hypothesis that maximizes $P(D \mid h)$ is called a maximum likelihood hypothesis (ML hypothesis). We don't have info about prior probabilities of the hypotheses so we assume they all have the same prior probability; the hypothesis space is uniformly distributed. If we assume $P(h_i) = P(h_j)\ \forall\, i, j \quad i \neq j$ then we can further simplify and choose the maximum likelihood (ML) hypothesis

$$h_{ML} = \underset{h \in H}{\operatorname{argmax}} \; P(D \mid h)$$

We can design a straightforward concept learning algorithm to output the maximum a posteriori hypothesis, based on the Bayes theorem:

- for each hypothesis $h \in H$, compute the posterior probability with Bayes $P(h \mid D) = \frac{P(D \mid h) P(h)}{P(D)} \quad \forall\, h$
- output the hypothesis h_{MAP} with the highest posterior probability $h_{MAP} = \underset{h \in H}{\operatorname{argmax}} \; P(h \mid D)$

We can implement this algorithm when it is feasible to enumerate every h in H.
This algorithm requires significant amounts of computation because it applies the Bayes theorem to each hypothesis h in H to compute $P(h \mid D)$. While this may prove impractical for large hypothesis spaces, the algorithm is still of interest because it provides a standard against which we may judge the performance of other concept learning algorithms.

Most probable classification ≠ Most probable hypothesis

h_{MAP} = most probable hypothesis given data D

Given a new instance x', what is its most probable classification?

$h_{MAP}(x')$ may not be the most probable classification

Consider three possible hypotheses h_1, h_2, h_3

$P(h_1|D) = 0.4, \quad P(h_2|D) = 0.3, \quad P(h_3|D) = 0.3$

Given a new instance x

$h_1(x) = \oplus \quad , \quad h_2(x) = \ominus \quad , \quad h_3(x) = \ominus$

x has probability 0.4 (40%) of being classified as a $\oplus$ and $0.3 + 0.3 = 0.6$ of being classified as $\ominus$

The most probable classification is $\ominus$ (60%)

The maximum a posteriori hypothesis is $h_{ML} = \underset{h \in H}{\operatorname{argmax}} P(h|D) = h_1$

$$h_{MAP} = \underset{h \in H}{\operatorname{argmax}} P(h|D) = \underset{h \in H}{\operatorname{argmax}} P(D|h)P(h)$$

$P(h)$ is constant $\rightarrow = \underset{h \in H}{\operatorname{argmax}} P(D|h)$

Bayes Optimal Classifier

Consider a target function $f: X \rightarrow V$, $V = \{v_1, \dots, v_k\}$, data set D and a new instance $x \notin D$

$$P(v_j|x, D) = \sum_{h_i \in H} P(v_j|x, h_i, D) P(h_i|x, D) = \sum_{h_i \in H} P(v_j|x, h_i) P(h_i|D)$$

- x is not in D (h_i is conditionally independent from D)
- each $P(v_j|x, h_i)$ is scaled down by $P(h_i|D)$, $\sum_i P(h_i|D) = 1$
- once we know h_i we don't need the dataset since it is derived from it

BOC $\rightarrow$ $$V_{OB} = \underset{v_j \in V}{\operatorname{argmax}} \sum_{h_i \in H} P(v_j|x, h_i) P(h_i|D)$$

Consider $H = \{h_1, h_2, h_3\}$

h_{MAP}

$P(h_1\|D) = 0.4$	$P(\ominus\|x, h_1) = 0$	$P(\oplus\|x, h_1) = 1$
$P(h_2\|D) = 0.3$	$P(\ominus\|x, h_2) = 1$	$P(\oplus\|x, h_2) = 0$
$P(h_3\|D) = 0.3$	$P(\ominus\|x, h_3) = 1$	$P(\oplus\|x, h_3) = 0$

Suppose a new instance x is encountered and classified as $\oplus$ by h_1 and as $\ominus$ by h_2, h_3

$$\sum_{h_i \in H} P(\oplus|x, h_i) P(h_i|D) = 0.4$$

$$\sum_{h_i \in H} P(\ominus|x, h_i) P(h_i|D) = 0.6$$

$\rightarrow$ the most probable classification ($\ominus$) is different from the classification generated by h_{MAP} (h_1)

The most probable classification is obtained by combining the predictions of all hypothesis, weighting them by the posterior probability of the generating hypothesis

$$V_{OB} = \underset{v_j \in V}{\operatorname{argmax}} \sum_{h_i \in H} P(v_j|x, h_i) P(h_i|D) = \ominus$$

This method is good but it is not applicable because we can't iterate over all hypotheses. We need another method $\rightarrow$ BOC is optimal but not applicable when H is huge or infinite

If $P(D, h)$ is a continuous function (continuous space of hypothesis leads to continuous probability distribution) we can find the argmax of $\log(P)$

$$h_{ML} = \underset{h_i \in H}{\operatorname{argmax}} P(D|h_i) = \underset{h_i \in H}{\operatorname{argmax}} (\log(P(D|h_i)))$$

The log turns the product (conditional independence $P(D|h) = \prod_i P(d_i|h)$) into sum

We have 5 kinds of bags containing 2 kinds of candies (l, c) in different proportions

10% of the bags are h_1 : 100% c, 0% l
20% of the bags are h_2 : 75% c, 25% l
40% of the bags are h_3 : 50% c, 50% l
20% of the bags are h_4 : 25% c, 75% l
10% of the bags are h_5 : 0% c, 100% l

We choose a random bag (without knowing which type it is) and extract some candies from it.
What kind of bag is it?
What is the probability of extracting a candy of a specific flavor next?

Prior probability distribution : $P(H) = \langle 0.1, 0.2, 0.4, 0.2, 0.1 \rangle$
likelihood for l candy : $P(l|H) = \langle 0, 0.25, 0.5, 0.75, 1 \rangle$
likelihood for c candy : $P(c|H) = \langle 1, 0.75, 0.5, 0.25, 0 \rangle$

What kind of bag is it?

$$P(h_i|D) = \alpha P(D|h_i)P(h_i) \quad \longleftarrow \text{Bayes rule}$$

We need a dataset. Suppose we extract 1 l candy

$$\left.\begin{aligned} P(h_1|d_1) &= \alpha P(d_1|h_1)P(h_1) \\ &= \alpha P(l|h_1)P(h_1) \\ &= \alpha \cdot 0 \cdot 0.1 \\ P(h_2|d_1) &= \alpha \cdot 0.25 \cdot 0.2 \\ P(h_3|d_1) &= \alpha \cdot 0.5 \cdot 0.4 \\ P(h_4|d_1) &= \alpha \cdot 0.75 \cdot 0.2 \\ P(h_5|d_1) &= \alpha \cdot 1 \cdot 0.1 \end{aligned}\right\} \quad P(H|D=\{d_i\}) = \alpha \langle 0, 0.25, 0.5, 0.75, 1 \rangle \cdot \langle 0.1, 0.2, 0.4, 0.2, 0.1 \rangle = \alpha \langle 0, 0.05, 0.2, 0.15, 0.1 \rangle$$

(only one extraction)

Suppose we extract another l candy

$D_2 = \{d_1, d_2\}$ subsequent extractions are independent from each other
Bayes + chain rule

$$P(h_i|\{d_1,d_2\}) = \alpha P(\{d_1,d_2\}|h_i)P(h_i)$$
$$= \alpha P(\{d_2\}|h_i) \underline{P(\{d_1\}|h_i)P(h_i)}$$

(underlined: previous step)

$$= \alpha \langle 0, 0.25, 0.05, 0.75, 1 \rangle \cdot \alpha \langle 0, 0.05, 0.2, 0.15, 0.1 \rangle = \dots$$

What is the probability of having another candy l after $D_3 = \{d_1, d_2, d_3\}$ = ?

$$P(l|D_3) = \sum_{h_i \in H} \underbrace{P(l|h_i)}_{\text{vote of each hypothesis for } l} \underbrace{P(h_i|D_3)}_{\text{weights of each hypothesis (how likely this hypothesis is)}} = \text{optimal prediction}$$

General approach
Given dataset $D = \{d_i\}$ with $d_i \in \{0, 1\}$
assuming a probability distribution $P(d_i, \theta)$ → parametric probability distribution parametrized by θ

$$\theta_{ML} = \underset{\theta}{\text{argmax}} \log P(d_i | \theta)$$

we can apply the same model to multiple distributions (e.g. Bernoulli)

Bernoulli distribution is the discrete probability distribution of a binary random variable $X \in \{0,1\}$

$$P(X = 1) = \theta \quad ; \quad P(X = 0) = 1 - \theta$$

(e.g. observing head after flipping a coin, extracting candy ℓ from a set containing 2 candies, ...)

$$P(X = k, \theta) = \theta^k (1 - \theta)^{1-k}$$

Multivariate Bernoulli distribution is the joint probability distribution of a set of binary random variables $X_1, \dots, X_n$, each of them following the Bernoulli distribution.

$$P(X_1 = x_1, X_2 = x_2, \dots, X_n = x_n, \theta_1, \theta_2, \dots, \theta_n) = \prod_{i=1}^{n} P(X_i = x_i, \theta_i) = \prod_{i=1}^{n} \theta^{k_i} (1 - \theta_i)^{1-k_i}$$

this under the assumption that random variables X_i are mutually independent → Multivariate Bernoulli is the product of n Bernoulli distributions

e.g. $\underbrace{\text{observing head after flipping a coin}}_{X_1 = \text{head}}$ AND $\underbrace{\text{extracting candy } \ell}_{X_2 = \ell}$

$$P(X_1 = \ell, X_2 = \ell, \theta_1, \theta_2) = \theta_1^{k_1} (1 - \theta_1)^{1-k_1} \cdot \theta_2^{k_2} (1 - \theta_2)^{1-k_2}$$

θ_1 = number of head observed / number of coin flip $\sim \frac{1}{2}$
θ_2 = number of ℓ candies / number of candies in the bag $\in [0, 1]$

Binomial distribution is the probability distribution of k outcomes from n Bernoulli trials

$$P(X = k, n, \theta) = \binom{n}{k} \underbrace{\theta^k (1 - \theta)^{n-k}}_{\text{Bernoulli}}$$

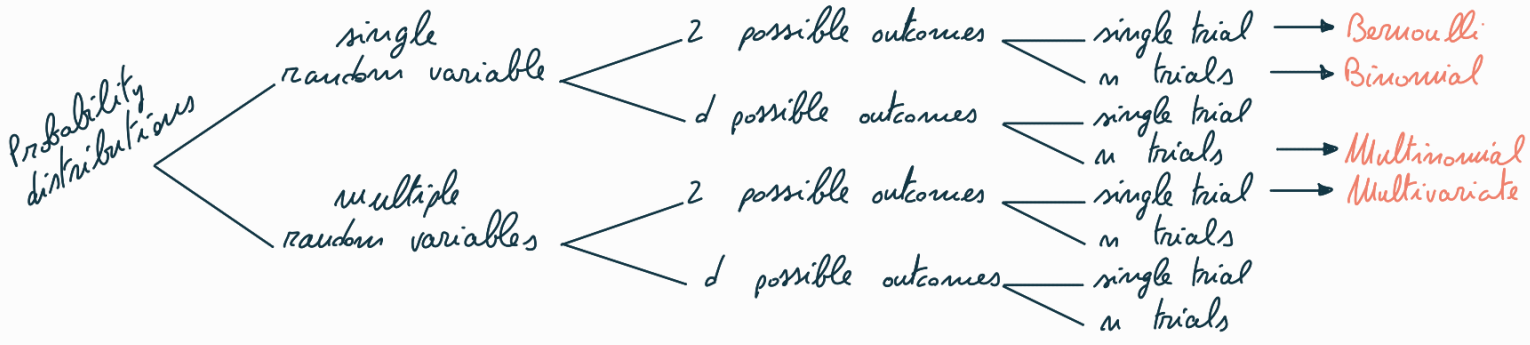
$\binom{n}{k}$ ← k outcomes over n trials

e.g. flipping a coin n times and observing k heads

Multinomial distribution is the generalization of binomial distribution for discrete valued random variables with d possible outcomes.
X_d has d possible outcomes $(k_1, \dots, k_d)$. What is the probability of observing a particular outcome k times after n trials?

$$P(X_1 = k_1, \dots, X_d = k_d, n, \theta_1, \dots, \theta_d) = \frac{n!}{k_1! \dots k_n!} \theta_1^{k_1} \cdot \dots \cdot \theta_d^{k_d}$$

e.g. rolling a d-sided die n times and observing k times a particular value
extracting k ℓ candies after n trials from a bag containing d different flavors



Bernoulli : single random variable , binary , single trial
Binomial : single random variable , binary , n trials
Multinomial : single random variable , multinomial , n trials
Multivariate : multiple random variables , binary , single trial

Probabilistic classification : $\underset{v_j \in V}{\operatorname{argmax}} P(v_j | x, D)$

Bayes Optimal Classifier provides best result, not practical when hypothesis space is large

Continuous model

Maximum likelihood estimation efficiently (number of hypotheses is small or there is a simple analytical solution) solved when analytical solutions are available

Naive Bayes Classifier is an approximation of the Bayes Optimal Classifier. It uses conditional independence to approximate the solution. It is used when each instance is described by a set of attributes : $f : A_1 \times A_2 \times \dots A_n \longrightarrow V = \{v_1, v_2, \dots, v_k\}$

$P(x, y | z) = P(x | y, z) P(y | z) = P(x | z) P(y | z)$ (the same without y) + $\underset{v_j \in V}{\operatorname{argmax}} P(v_j | x, D) = \underset{v_j \in V}{\operatorname{argmax}} P(v_j | a_1, a_2, \dots, a_n, D)$

X is conditionally independent of Y given Z | BOC with x described as a set of features

$$v_{MAP} = \underset{v_j \in V}{\operatorname{argmax}} P(v_j | a_1, a_2, \dots, a_n, D) = \operatorname{argmax} \frac{P(a_1, a_2, \dots, a_n | v_j, D) P(v_j | D)}{\cancel{P(a_1, a_2, \dots, a_n | D)}} \text{ normalization factor}$$

$$= \operatorname{argmax} P(a_1, a_2, \dots, a_n | v_j, D) P(v_j | D) = \underset{v_j \in V}{\operatorname{argmax}} P(v_j | D) \prod_i P(a_i | v_j, D)$$

(conditional independence assumption)

The Naive Bayes assumes that all attributes are mutually conditionally independent from each other given the dataset and the classification value ; this is not real in general but we are approximating the solution ; it is therefore no more guaranteed to get the optimal value. If we don't make this assumption the joint probability $P(a_1, a_2, \dots, a_n)$ will be exponential in the number of attributes and will make the system impractical. Conditional independence can bring exponential systems to linear.

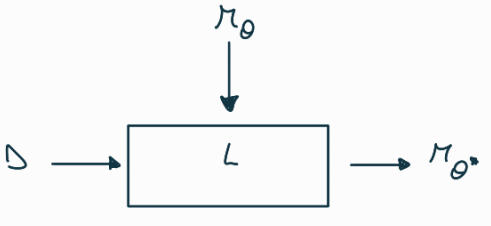
$$v_{NB} = \underset{v_j \in V}{\operatorname{argmax}} P(v_j | D) \prod_i P(a_i | v_j, D)$$

This is a practical method

(implicit dependency on the dataset)

Naive Bayes Algorithm

$f : A_1 \times A_2 \times \dots \times A_n \longrightarrow V = \{v_1, v_2, \dots, v_k\}$
dataset D
new instance $x = \langle a_1, a_2, \dots, a_n \rangle$
} inputs




```
Naive_Bayes_learner (A, V, D) :
    for each target value v_j ∈ V
        P̂(v_j | D) ← estimate P(v_j | D)
        for each attribute A_i ∈ A
            for each attribute value a_i ∈ A_k
                P̂(a_i | v_j, D) ← estimate P(a_i | v_j, D)
```

now we have $\hat{P}(v_j | D)$ and $\hat{P}(a_i | v_j, D)$ for each attribute value for each attribute

Classify_new_instance (x) : $v_{NB} = \underset{v_j \in V}{\operatorname{argmax}} \hat{P}(v_j | D) \prod_{a_i \in x} \hat{P}(a_i | v_j, D)$

e.g. playtennis example

Day	Outlook	Temperature	Humidity	Wind	PlayTennis		
1	Sunny	Hot	High	Weak	No	-	w
2	Sunny	Hot	High	Strong	No	-	s
3	Overcast	Hot	High	Weak	Yes	+	w
4	Rain	Mild	High	Weak	Yes	+	w
5	Rain	Cool	Normal	Weak	Yes	+	w
6	Rain	Cool	Normal	Strong	No	-	s
7	Overcast	Cool	Normal	Strong	Yes	+	s
8	Sunny	Mild	High	Weak	No	-	w
9	Sunny	Cool	Normal	Weak	Yes	+	w
10	Rain	Mild	Normal	Weak	Yes	+	w
11	Sunny	Mild	Normal	Strong	Yes	+	s
12	Overcast	Mild	High	Strong	Yes	+	s
13	Overcast	Hot	Normal	Weak	Yes	+	w
14	Rain	Mild	High	Strong	No	-	s

classification value

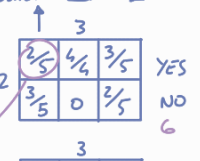
conditional probability $P(O=s, PT)$

sum of column $\sum = 1$

$P(O \mid PT)$

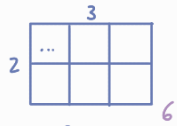
Outlook = { Sunny, Overcast, Rain }

5 entries have $O=s$



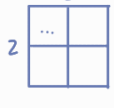
$P(H \mid PT)$ 2 of which are YES

Humidity = { Hot, Mild, Cold }



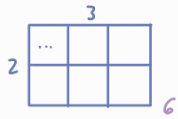
$P(W \mid PT)$

Wind = { Weak, Strong }



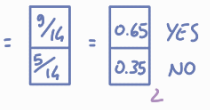
$P(T \mid PT)$

Temperature = { Hot, Mild, Cold }



PlayTennis = { YES, NO }

$P(v_j \mid D) =$ 9/14 = 0.65 YES; 5/14 = 0.35 NO

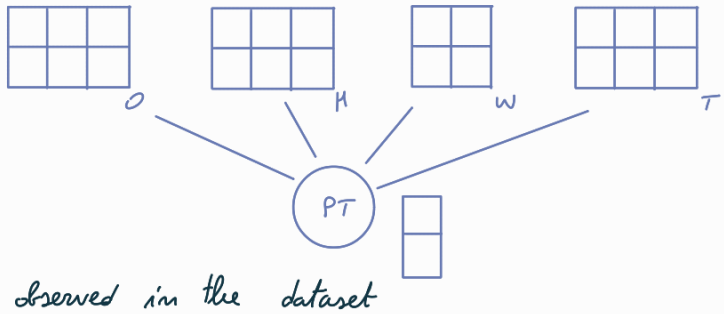


The model will be complete when we'll have these $6+6+4+6+2 = 24$ numbers

We didn't account for the fact that the sum of each column is 1. The number of trainable parameters (size of the model) is exactly half since we only have the freedom to choose 1 row (the other one is complementary). This is a special case, not always the size is exactly half.

Size of the model $= 24/2 = 12$

We can consider the problem as a Bayesian network →



What if a $P(a_i \mid v_j, D)$ is 0 ?

$P(A_2 = a_2 \mid v_j, D) = 0 \longrightarrow \Pi(\cdot) = 0$

This happens if an attribute is never observed in the dataset

e.g. never observed in the dataset that the output is NO when Outlook is Overcast

Typical solution is Bayesian estimate with prior estimates

$$\hat{P}(v_j \mid D) = \frac{|\{\langle \dots, v_j \rangle\}|}{|D|} = \frac{\text{times class } v_j \text{ is observed in } D}{\text{number of elements in } D}$$

$$\hat{P}(a_i \mid v_j, D) = \frac{|\{\langle \dots, a_i, \dots, v_j \rangle\}|}{|\{\langle \dots, v_j \rangle\}|} = \frac{\text{times attribute } A_i \text{ has value } a_i \text{ when class } v_j \text{ is predicted}}{\text{times class } v_j \text{ is observed in } D}$$

$$\frac{|\{\langle \dots, a_i, \dots, v_j \rangle\}| + mp}{|\{\langle \dots, v_j \rangle\}| + m}$$

mp → p is a prior estimate for $P(a_i \mid v_j, D)$

m → m is a weight given to prior

new instance $x = \langle$ Outlook = s., Temperature = c., Humidity = h., Wind = s. $\rangle$

$$V_{NB} = \underset{v_j \in [YES, NO]}{\operatorname{argmax}} \; P(v_j) \overset{\wedge}{\prod} P(a_i \mid v_j)$$

N.B. conditioning on D is omitted

$$= \underset{v_j \in [YES, NO]}{\operatorname{argmax}} \; P(v_j) \, P(\text{Outlook} = s. \mid v_j) P(\text{Temperature} = c. \mid v_j) \, P(\text{Humidity} = h. \mid v_j) \, P(\text{Wind} = s. \mid v_j)$$

Conditional independence assumption is often violated